分散処理システムにおけるプロセスの割り当て問題に対する近似解法(待ち行列理論とその周辺)

AUTHOR(S):

下条, 真司; 宮原, 秀夫; 高島, 堅助

CITATION:

下条, 真司 ...[et al]. 分散処理システムにおけるプロセスの割り当て問題に対する近似解法(待ち行列理論とその周辺). 数理解析研究所講究録 1986, 596: 141-154

ISSUE DATE:

1986-07

URL:

http://hdl.handle.net/2433/99548

数理解析研究所講究録
596巻 1986年 141-154

分散処理システムにおけるプロセスの割り当て問題に対する近似解法

大阪大学基礎工学部情報工学科　下条真司　(Shinji Shimojo)
宮原秀夫　(Hideo Miyahara)
高島堅助　(Kensuke Takashima)

1．まえがき

ここでは、分散処理システムを図1のように考える．処理要素(PE: Processing Element)は、単一のプロセッサと独自のローカル・メモリー(LM: Local Memory)からなる．各ＰＥは相互通信網(CN: Communication Network)を通じて互いに結合されている．ＣＮとしては共通バスやクロスバー・スィッチのようなものから、ＬＡＮのようなものまで種々考えられる．

これら分散処理システムで処理される一まとまりの仕事をジョブと呼ぶ．ジョブは複数のプロセスからなり、これらのプロセスは各々別々のＰＥで実行することができる．しかし、同一のジョブに属するプロセスは互いに関連性があり、プロセス間通信(IPC: Inter-Process Communication)によって互いのデータの授受、同期などを行う．

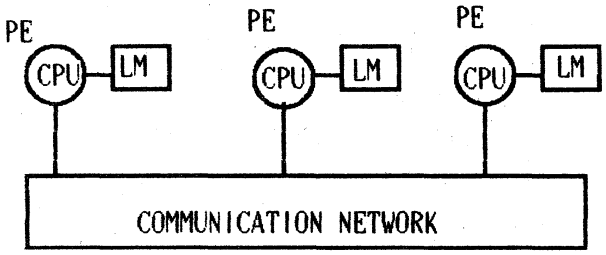


図1　分散処理システム

各ＰＥは同一の処理能力を持ち、プロセスの実行やＩＰＣを管理するＯＳを備えているものとする．一つのＰＥには複数のプロセスを割り当てることが可能であり、これらのプロセスはプロセッサ・シェアリング（ＰＳ）によって処理される．同一のＰＥに割り当てられたプロセス同志はＬＭを通じてＩＰＣを行うが、異なるＰＥに割り当てられたプロセスはＣＮを通じてＩＰＣを行い、これが分散化のオーバーヘッドとなる．この分散化によるオーバーヘッドはプロセスの並列性とトレードオフを持つ．つまり、並列性を大きくし、多くのＰＥを用いようとすると分散化のオーバーヘッドは大きくなる．一方、この分散化オーバーヘッドを軽減して少ないＰＥを用いようとすると、並列性は減少する．したがって、分散処理システムの構築における大きな問題は互いに依存性のあるプロセスをどのようにシステム上に分散するかということである．ここでは、これをプロセス割り当て問題と呼ぶ．

このようなプロセス割り当て問題は主として、組み合わせ最適化の手法を用いて

扱われて来た[4]、[5]．しかし、これらのアプローチではシステムを動的に（例えば、ＩＰＣにおける遅延時間が確率的に変化する場合など）扱うことができない．そこで、我々は先に文献[2],[3]において、一つのジョブに属するすべてのプロセスが終了する時間の平均を平均ジョブ実行時間として捉え、分散処理システムを待ち行列網理論を用いてモデル化することにより、これを解析的に求める方法を示した．また、この平均ジョブ実行時間を最小にするプロセス割り当て問題を考え、プロセスの平均実行時間と他のプロセスへの通信確率が等しい場合の割り当て問題に対して、最適プロセッサ数を求めた．

本論文では、分散処理システムのモデル化の際に [2],[3]では考慮していなかったＩＰＣとＰＥ内部での処理の重ね合わせの効果を導入している．[2],[3]ではあるＰＥがＩＰＣを行っているときにはそのＰＥに割り当てられている他のプロセスの処理を行わないとしたが、ここではこれを許した．このモデルから待ち行列網理論を用いて平均ジョブ実行時間を解析的に求めた．さらに、各プロセスの平均ＩＰＣ間隔、平均ＩＰＣ回数およびプロセス間通信確率が異なる一般の場合についての割り当て問題について、組み合わせ最適化の手法を用いた近似解法を示している．最後に数値例により近似解法の妥当性を論じている．

２．プロセスと分散処理システムのモデル

一つのジョブの全プロセス数をMとし、プロセスｉがＩＰＣを行ったときに、プロセスｊと通信する確率をp_{ij}とする．このとき、

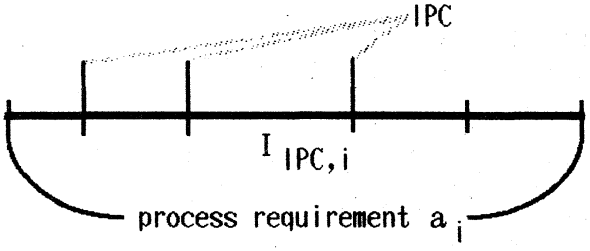


図1　プロセスｉのパラメータ間の関係

$$\sum_{i=1}^{M} p_{ij} = 1 \qquad i=1...M \qquad (1)$$

とする．したがって、プロセスｉにおいて次の値が知られているとする（図1）．

- 平均処理時間　I_i
- 平均IPC回数　a_i
- 平均IPC間隔　$I_{IPC,i}$

ただし、これら三つの量の間には

$$I_i = I_{IPC,i} \times a_i \qquad (2)$$

という関係があるとする．したがって、プロセスｉにおけるIPCから次のIPCまでの平均処理時間（１／$\mu_{CPU,i}$）は

$$1/\mu_{CPU,i} = I_{IPC,i} \qquad (3)$$

となる．LMまたはCNを通じてのIPCに必要な時間はそれぞれ平均１／μ_{lm}、１／μ_{cn}の指数分布に従うとする．あるプロセスがIPCを行っている間、CPUは他のプロセスの処理を行うことができる．つまり、IPCは内部IPC、外部IPC共にCPUにおける処理とオーバーラップして行われる．パラメータをまとめると表1のようになる．

表1　分散処理システムパラメータ

プロセス数	M
プロセス間通信確率	pij
平均ＩＰＣ回数	ai
割り当て行列	xij
LMを介しての平均ＩＰＣ時間	1/μlm
CNを介しての平均ＩＰＣ時間	1/μcn
CPU処理時間	1/μcpu,i

２．１　分散処理システムの待ち行列網によるモデル化

ある割り当てＸに対するPE数をn_p、PEｊに割り当てられたプロセス数m_jとすると、

$$m_j = \sum_{i=1}^{M} x_{ij} \qquad j=1...n_p \qquad (4)$$

ここで分散処理システムを閉じた待ち行列網としてモデル化する．各PEにおいて、CPUをPSの指数サーバー、LMをFCFSの指数サーバーとする．PEｊにおいてプロセスｉが内部IPCを行う確率$p_{I,i}$は（以下、プロセスｉの割り当てられているPEｊは一意に決定できるので特に必要ない限りはｊを省略する）、

$$p_{I,i} = \sum_{k=1}^{M} p_{ik} x_{ij} x_{kj} \qquad \text{for } i=1...M \qquad (5)$$

一方、PEｊにあるプロセスｉが外部IPCを行う確率$p_{E,i}$は

$$p_{E,i} = \sum_{k=1}^{M} p_{ik} x_{ij} (1 - x_{kj}) \qquad \text{for } i=1...M \qquad (6)$$

となる．(1)式より、$p_{I,i} + p_{E,i} = 1$である．プロセスｉが処理を終えてシステ

ムから出て行く確率$q_{1,i}$および処理の続行を受ける確率$q_{2,i}$は

$$q_{1,i} = 1 / a_i \qquad i=1...M \qquad (7)$$

$$q_{2,i} = 1 - q_{1,i} \qquad i=1...M \qquad (8)$$

したがって、分散処理システムの待ち行列網モデルは図2のようになる（プロセス同時到着部分については次節で説明する）．ここでは分散処理システムに割り当てられるプロセスを客と考え、各々の客が別々のチェインをなすとして、単一のチェインだけを示している．PEjにおけるCPUおよびLMはそこに割り当てられたm_j個のプロセスによって共有され、CNはM個の全プロセスによって共有されている．

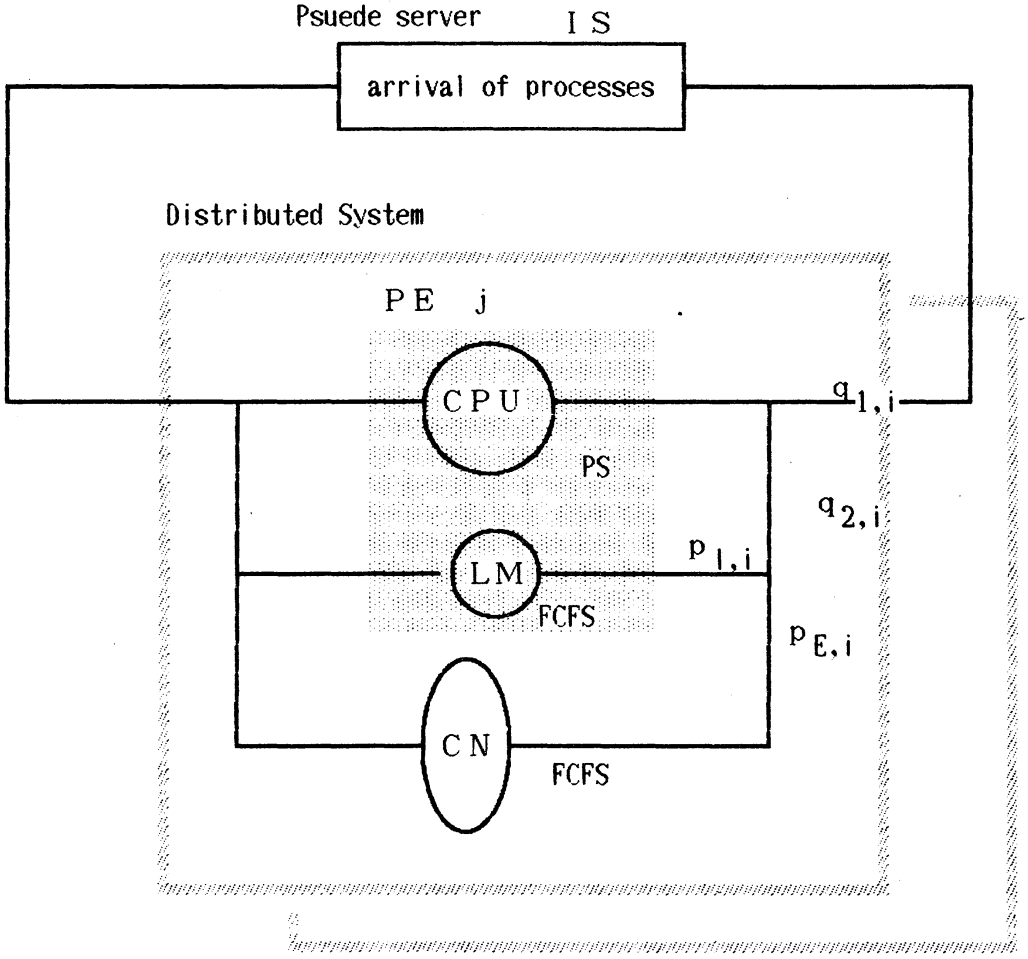


図2　分散処理システムの待ち行列モデル

2．2．　プロセスの同時到着

一つのジョブに属するM個のプロセスは実際には同時に図2の分散処理システム内に到着し、並行に処理されシステムから出る．ジョブの終了は最後のプロセスがシステムから退去することによって起こる．このような特殊な客の分散処理システムにおける応答時間を求めるために、[6]で用いられている疑似サーバーの手法を導入した．

プロセスの同時到着は分散処理システムの外に疑似サーバーを置くことにより実現される．すなわち、到着において、各プロセスはこの疑似サーバーから同時に分散処理システムに到着する．処理を終え分散処理システムから退去したプロセスは一旦疑似サーバーに入り、他のすべてのプロセスが終了するまで待たされる．最後のプロセスが処理を終え疑似サーバーに入ると、再びすべてのプロセスが分散処理システム内に入ることができる．今、プロセスｉの分散処理システムにおける在中時間を確率変数R_iで表すと、すべてのプロセスの終了時間すなわちジョブ実行時間D_0は、

$$D_0 = max(R_1, R_2 \ldots, R_M) \qquad (9)$$

で与えられる．ここで、$R_1, R_2 \ldots, R_M$が独立な平均$r_1, r_2 \ldots, r_M$の指数分布に従う(*)とすれば、D_0の期待値、d_0、は

$$\begin{aligned} d_0 = E(D_0) = & \sum_{i=1}^{M} 1/r_i - \sum_{i<j} 1/(r_i + r_j) \\ & \ldots \\ & + (-1)^{M-1} \sum_{i_1<i_2<\ldots<i_M} 1/(r_{i1} + r_{i2} + \ldots r_{iM}) \end{aligned} \qquad (10)$$

となる．したがって、プロセスｉの疑似サーバーでの待ち時間の期待値d_iは

$$d_i = d_0 - r_i \qquad i=1\ldots M \qquad (11)$$

となる．疑似サーバーのサービス規約を無限サーバー(IS)とすると、図2はBCMPクラスの閉じた待ち行列網となり、ISサーバーにおける待ち時間の初期値d_i(i=1...M)を適当に与えれば、図2の待ち行列網モデルを解くことにより、プロセスの平均在中時間r_i(i=1...M)を求めることができる．すると、(10),(11)式より新たなd_i

---

(*)　CNを通してIPCを行う確率が小さい場合には、各プロセスのシステム在中時間はほぼ独立と考えられる．また、R_iが独立でない場合、(10)式は最大値を与えていると思われる．

(i=1...M)が求まる．このようにして、繰り返しにより平均ジョブ実行時間d_0を求めることができる（繰り返しの収束性については厳密ではないが、簡単な証明が[6]に示されている）．

3．　プロセス割り当て問題の近似解法

ここでは、IPC確率p_{ij}、平均IPC回数a_iおよび平均IPC間隔$1/\mu_{cpu,i}$がプロセスごとに異なる一般の場合におけるプロセス割り当てについて考える．この場合に先に述べたジョブ実行時間の平均値を目的関数とし、これを最小にするプロセス割り当てを見出すことはきわめて困難である．そこでここでは新たに次のような割り当て問題を考える．

(H.1)　PEに割り当てられるプロセスの平均IPC回数a_iの総和に、すべてのPEに対し共通の上限dを設ける．

すなわち、

$$\sum_{i=1}^{M} a_i x_{ij} \leq d \quad \text{for} \quad 1 \leq j \leq n_p$$

これを満たす下で、

(H.2)　異なるPEに割り当てられたプロセス間のIPCを最小化する．

PEに割り当てられたプロセスの平均IPC回数の総和に共通の上限dを設けることにより、単一のPEにプロセスが集中するのを制限することができる．つまり、各PEごとにロードバランスが実現できることになる．上記の割り当て問題から求められた割り当てを元の平均ジョブ実行時間を最小にする割り当てに対する近似解として採用する．この問題は比較的簡単な整数計画問題として定式化でき、分枝限定法を用いることにより元の問題よりはるかに簡単に解くことができる[9]．

したがって、ここではまず、(H.1)，(H.2)を同時に満たす割り当てを求める問題を整数計画問題として定式化する．次にこの整数計画問題に対する最適解を分枝限定法を用いて求める手法を示す．さらに近似解を改良する手法について述べる．

3．1　割り当て問題の定式化

まず、この場合のプロセス割り当て問題を0/1整数計画問題として次のように定式化する[1]．

変数

$x_{ij} = 0$　　プロセスiをPEjに割り当てる．

1　　その他　　(12)

目的関数

$$\min \quad f(X) = \sum_{k<l} \sum_{i \neq j} \left(\sum \sum t_{kl} x_{ki} x_{lj} \right) \qquad (13)$$

制約条件

x_{ki}は0／1変数 (14)

$$\sum_{i=1}^{n_p} x_{ij} = 1 \quad \text{for } 1 \leqq i \leqq M \qquad (15)$$

$$\sum_{i=1}^{M} a_i x_{ij} \leqq d \quad \text{for } 1 \leqq j \leqq n_p \qquad (16)$$

但し、

$$t_{kl} = \begin{cases} p_{kl} / \mu_{cpu,k} + p_{lk} / \mu_{cpu,l} & k < l \\ 0 & \text{その他} \end{cases} \qquad (17)$$

d：PEあたりの平均ＩＰＣ回数の総和の上限

(17)式のt_{kl}はプロセスkとプロセス1の間の単位時間あたりのプロセス間通信確率を表している．ここで、p_{kl}はIPCあたりであるので、これを平均IPC間隔1／$\mu_{cpu,k}$で正規化している．このようにすると目的関数(13)により、(H.2)すなわち、プロセス間の通信量の総和を最小にすることが実現できる．dを変えることにより、プロセスのPEへの集中度を調節することができる．dの決定方法については後述することとし、以下dが与えられたとして議論を進める．

３．２　分枝限定法による解の探索

(12)-(17)の0/1整数計画問題を分枝限定法を用いて解く．分枝限定法における解の探索において、プロセスを一回に一つづつ割り当てて行く．探索は深さ優先で行う．例えば三つのプロセスを二つのPEに割り当てるとすると探索木は図3のようになる．ここで木の頂点は割り当てを行うプロセスを、木の枝がそのプロセスを割り当てるPEを示している．中間の頂点は一部のプロセス割り当てのみが完了した状況を現しており、これを部分割り当てと呼ぶ．

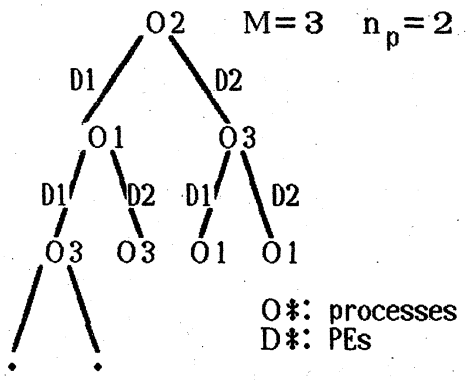


図3　割り当ての探索木

一方、葉頂点はすべてのプロセスを割り当てた一つの実行可能解を現し、これを完全割り当てと呼ぶ．プロセス割り当てはこのような木をたどって行くことにより、以下のように進んで行く．

1） 割り当てるプロセスの選択
まだ、割り当てられていないプロセスを一つ選ぶ．このプロセスをｉとする．
2） PEの選択
まだ、テストされていないPEを選ぶ．このPEをｊとする．
3） プロセスｉをPEｊに割り当てた後の部分割り当てＶによる目的関数値の下限ｇ（Ｖ）を計算し、完全割り当てによって現在までに得られた最良の目的関数値ｚと比較する．

$$z \leqq g(V) \qquad (18)$$

ならば以下の探索を中止し、別の選択を行う．

３．３ 発見的手法

探索において候補となるPEおよびプロセスの選択に発見的手法を導入することにより最適解を早く求めることが期待できる．そのため、PE、プロセスの選択は次のように行う．

(H.3) プロセスの選択
まだ割り当てられていないプロセスのうち、

$$a_i = \max_k (a_k) \qquad (19)$$

すなわち、平均IPC回数が最大となるプロセスｉを選ぶ．

(H.4) PEの選択

$$\max_j \left(\sum_{l \in S_j} t_{il} \right) \qquad (20)$$

なるPEｊを選ぶ．ここで、

S_j＝｛ｋ｜ｋ≠ｉ、部分割り当てX_iのうちPEｊに割り当てられているプロセスの添字の集合｝

すなわち、割り当てようとするプロセスｉと現在までにPEｊに割り当てられているプロセスとの通信量の総和が最大となるPEｊを選ぶ．

(H.3)はビンパッキング問題のFFD(First Fit Decreasing)アルゴリズムに相当し、

ロードバランスを実現しながら、使用するPE数を最小にできる[1]．また、(H.2)を適用することにより、プロセスｉを割り当てることによる目的関数値の上昇を最小限に押さえることができる．

３．４　目的関数値の下限の導出

現在の部分割り当てより派生するすべての完全割り当てに対する目的関数の下限を計算することにより、探索する範囲を限定することができる．部分割り当てＶに対する目的関数値の下限として

$$g(V) = f(V) + \sum_{i \in U} \left(\sum_{j} \sum_{l \in S_j} t_{il} - \max_{j} \left(\sum_{l \in S_j} t_{il} \right) \right) \qquad (21)$$

を用いる．ここで、

Ｕ＝｛ｉ｜まだ割り当てられていないプロセスの添字の集合｝

すなわち、(16)式の制限を考えずに(H.4)を適用して、残りのすべてのプロセスを割り当てたときの各プロセスによる目的関数の増加量と部分割り当てＶによる目的関数値の和である．

３．５　ｄの決定方法

ここまでｄの値は、与えられたものとして進めて来た．ここでは各PEに割り当てられるプロセスの平均ＩＰＣ回数の総和の上限であるｄを定める方法について述べる．(12)-(17)の整数計画問題は与えられたｄに対し通信量最小の割り当てを求める．ｄの範囲は次のようになる．

$$d_{min} = \max_{i}(a_i) \leqq d \leqq \sum_{i} a_i = d_{max} \qquad (22)$$

すなわち、ｄは最小値d_{min}において最大数のPEを使用し（＝Ｍ）、最大値d_{max}において最小数のPEを使用する（＝１）．(22)内のそれぞれのｄの値について先の分枝限定法により整数計画問題(12)-(17)を解くことができ、ロードバランスを実現しながら外部通信量最小の割り当てを得ることができる．ただし、このようにして得られた割り当てはあくまでも近似解であり、平均ジョブ実行時間を最小にするとは限らない．したがって、いくつかの適当なｄに対して整数計画問題(12)-(17)を解き、これらの割り当てのそれぞれに対し、2章の待ち行列網モデルを適用して平均ジョブ実行時間を計算し、それらの中から最小の平均ジョブ実行時間を与えるものを採用することにより、近似解を改良することができる．

したがって、一般的なプロセス割り当て問題の近似解法を以下のようにまとめる

ことができる．

1) 調べるｄの数をｎとする．ｄのきざみΔｄを

$$\Delta d = (d_{min} - d_{max}) / n \qquad (23)$$

とする．ｄの初期値をｄ＝d_{min}とする．
2) 与えられたｄに対して整数計画問題(12)-(17)を上記の分枝限定法を用いて解く．
3) 2)で求まった割り当てに対し、2章の待ち行列網モデルを適用し平均ジョブ実行時間を求める．
4) ｄ＝ｄ＋Δｄ
5) 2)-4)をｄ＝d_{max}まで繰り返し、平均ジョブ実行時間を最小にする割り当てを近似解として採用する．

ｎを大きくすることにより調査する候補数が多くなり、近似解は良くなるが計算時間は増大する．したがって、要求される近似解の精度に対して適当なｎの値を選ぶことができる．

4． 考察

3章で示した一般の場合のプロセス割り当て問題に対する近似解法の得失を論じる．ここでは、すべてのプロセスの平均IPC間隔が等しいと仮定し（$1/\mu_{cpu,i} = 1/\mu_{cpu}$ for i=1...M）、プロセスｉがプロセスｊとIPCを行う確率は、平均0.5、標準偏差0.5の正規分布より発生させ、(1)式を満たすように正規化される．また、プロセスの要求処理量a_iは平均20、標準偏差20または2の正規分布より発生する．分散処理システムのパラメータは以下のとおり．

$\mu_{cpu} = 0.1$
$\mu_{lm} = 0.5$
$\mu_{cn} = 0.15$または0.5

近似解法において探索する候補の数ｎを20とした．結果は分散処理システムがIPC依存型（$\mu_{cn} = 0.15$）である場合と内部処理依存型（$\mu_{cn} = 0.5$）である場合の二つについて行う．IPC依存型の場合、CNにおける処理時間がLMにおけるそれと比べて大きいため少ないPEを用いた割り当てが望まれ、一方要求処理量依存型の場合、CNにおける処理時間とLMにおけるそれが等しいため負荷をできるだけ分散させることが望まれる．さらに、それぞれの場合について、平均IPC回数の分散が大きい場合（高分散）と小さい場合（低分散）を比べ、平均IPC回数の分散の影響を考察する．高分散の場合、負荷を均等に分散するのが難しくなり、一方低分散の場合、均

質なプロセスの割り当て問題により近づく．

表2にＭ＝5における3章で述べた近似解法および厳密解によって求めた割り当てに対する平均ジョブ実行時間と探索木における訪問頂点数を示す．厳密解は平均ジョブ実行時間を目的関数として分枝限定法により求めた．表から近似解法によって求めた割り当てはIPC依存型で平均ジョブ実行時間の分散が大きい場合でも厳密解で求めた割り当てによる平均ジョブ実行時間に近い解が得られていることがわかる．低分散の場合および内部処理依存型の場合には近似解法によって最適解が得られている．表2には参考のためにｄによる制限を加えない場合の分枝限定法による訪問頂点数の最悪値を示している．近似解法においてはｎ＝20回も探索を繰り返しているにもかかわれらず、ｄの上限および下限による限定操作により訪問頂点数が相当少ないことがわかる．厳密解では訪問頂点数はかなり少ないが一つの頂点を訪問する度に2章で示した待ち行列網モデルを解く必要があり、計算時間では近似解法のほうがははるかに少ない．例えば、Ｍ＝5で、IPC依存型高分散の場合、VAX750/unix上で近似解法46秒に対して厳密解は5分33秒かかる．低分散の場合で近似解法1分15秒に対して厳密解は約7分かかっている．表3にはＭ＝8の場合の近似解法による結果を示している．この場合、近似解法による訪問ノード数の削減効果はＭ＝5に比べてはるかに大きく、Ｍが大きくなるほど効率がよくなっていくことが分かる．また、平均IPC回数の分散が大きいほど上限ｄのために限定されるノード数は多い．計算時間は低分散の場合でもVAX750/unix上で2～3時間程度ですむが、厳密解を求めるのはほとんど不可能である．また、比較のために（）内にランダム・プロセス割り当てを行ったときの平均ジョブ実行時間を示している．これはランダムな10個の割り当てから求めた平均である．これから、ランダムに割り当てるよりも近似解

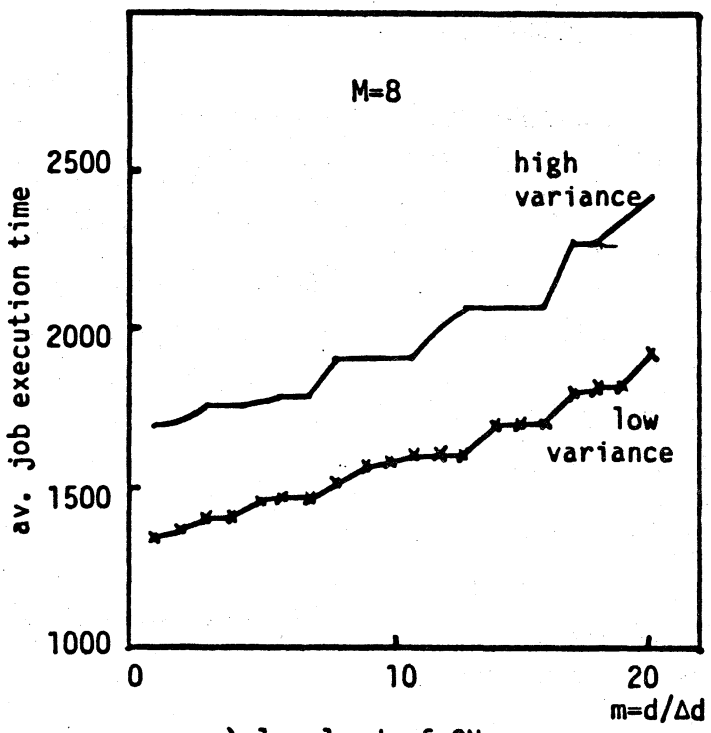
b）内部処理依存型

		高分散	低分散
近似解	平均ジョブ実行時間	984.9	544.7
	訪問ノード数	1597	2790
厳密解	平均ジョブ実行時間	984.9	544.7
	訪問ノード数	12	50
最悪訪問ノード数		3125	

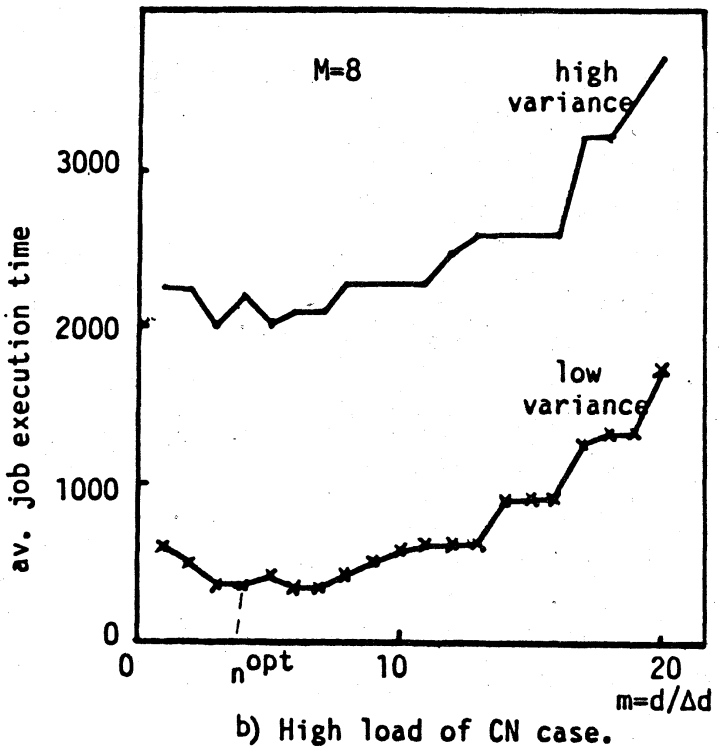
表3　平均ジョブ実行時間（Ｍ＝8）

	高分散	低分散
ＩＰＣ依存型		
平均ジョブ実行時間	2000.4 (2229.9)	1176.4 (1265.3)
訪問ノード数	35697	221641
内部処理依存型		
平均ジョブ実行時間	1400.4 (1543.4)	719.2 (889.9)
訪問ノード数	35697	221641
最悪訪問ノード数	16777216	

()はランダム割り当てによる平均ジョブ実行時間



a) low load of CN case.



b) High load of CN case.

図4. 平均ジョブ実行時間の上限dに対する変化

法を用いることによって2割程度平均ジョブ実行時間が改良されていることがわかる．その効果はIPC依存型のようにCNに対する負荷が高いほど、また、各プロセスの平均IPC回数の分散が大きいほど高い．　図4-a,bにはM＝8、内部処理依存型、IPC依存型それぞれの場合の(16)式の平均IPC回数の上限の効果を表している．つまり、図は各PEに割り当てられるプロセスの平均IPC回数の総和の上限dを(22)式の範囲すなわち、d_{min}からd_{max}までΔdずつ変化させて各dの値について得られた近似割り当てに対する平均ジョブ実行時間を示している．ただし、横軸は各dをΔdで正規化した値m、すなわち、

$$m = d / \Delta d \qquad d_{min} \leqq d \leqq d_{max} \qquad (24)$$

を示しており、n＝20と(23)よりこの場合$0 \leqq m \leqq 20$である．dを増加させていくと、外部IPCを最小にすることを目的関数としているため、プロセスの割り当てられるPE数は減少して行く．内部処理依存型の場合（図4-a）、CNに対する外部 IPCによる負荷が小さいためにPE数の大きいところ（dの小さなところ）で平均ジョブ実行時間は最小になる．特に、低分散の場合、各PEに一つづつプロセスを割り当てたものが最適解となっている．一方、外部IPC依存型の場合（図4-b）内部処理依存型よりも少ないPE数（dの大きなところ）で平均ジョブ実行時間は最小になる．このとき低分散の場合の最適PE数はn_p＝4である．探索する候補数nを増やせばそれだけ近似解を厳密解に近づけることができるが、計算時間も増加する．したがって、できるだけnを少なくすること、あるいは候補を調べる間隔Δdをできるだけ大きくとることが望まれる．図がdの変化に対して比較的滑らかであることからΔdを2～3倍して調べる候補数を減らしても、そこから得られる最適解はそれほど悪くならない．したがって、nの適当な値としては6～10程度で充分であることが言える．また、平均ジョブ実行時間の変化はdに対してほぼ凸であり、mを増やしながら局所最適解が求まった時点で探索を打ち切ってもよいことがわかる．

５．むすび

分散処理システムに対してＩＰＣとＣＰＵの処理のオーバーラップを含めた新しいモデルを提唱し、待ち行列網理論を用いて平均ジョブ実行時間を解析的に求めた．この結果をもとに分散処理システムにおけるプロセス間通信確率、平均ＩＰＣ回数および平均ＩＰＣ間隔が異なる一般の場合のプロセス割り当て問題に対し、分岐限界法を用いた近似解法を示し、その得失を明らかにした．

その結果、目的関数として直接平均ジョブ実行時間を用いるのではなく、プロセス間ＩＰＣ確率を用い、さらにロードバランスの制約を加えることにより、計算時間も早くよい近似解を求めることができた．

参考文献

（１）茨木俊秀：”組合せ最適化”、産業図書.
（２）下条、西田、宮原、高島、“通信するプロセスの割り当て問題に対する一考察”、昭５９年信学総全大後期、１Ｃ－８.
（３）下条、宮原、高島、“通信競合を含めたマルチプロセッサにおけるプロセス割り当て問題”信学論（Ｄ）Ｊ６８－Ｄ、５、pp.1049-1056（昭６０年５月）.
（４）A. Gabrieliam and D. B.Tyler: "Optimal Object Allocation in Distributed Computer Systems." 1984.
（５）C. Shen and W. Tsai: "A Graph Matching Approach to Optimal Task Assignment in Distributed Computing Systems Using a Minmax Criterion," IEEE Trans. on Computer Vol. C-34, NO. 3, MAR 1985.
（６）P. Heidelberger and K. S. Trivedi: "Analytic Queueing Models for Programs with Internal Concurrency," IEEE Trans. on Computers, Vol. C-32, No. 1, JAN 1983.